Broadview®
www.broadview.com.cn

自己动手写Docker

陈显鹭　王炳燊　秦妤嘉◎著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

作者简介

陈显鹭

阿里云高级研发工程师，对 Docker 有深入研究，是 Docker 多个项目的 Contributor，专注于容器技术的编排与基础环境研究。爱好折腾源代码，热爱开源文化并积极参与社区开源项目的研发。

王炳燊

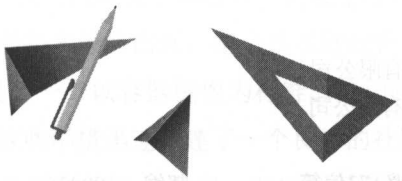
阿里云研发工程师，具有丰富的 Linux 开发经验，对 Docker 有深入研究，多次提交 Docker Patch。目前从事阿里云容器服务网络方案的设计与实现，专注于容器技术的基础环境研究。

秦妤嘉

阿里云高级研发工程师、DevOps 工程师，有丰富的容器化持续集成和持续交付开发实战经验，进行过 Jenkins 源码分析改造和 Jenkins 插件开发。目前从事阿里云容器服务持续集成和持续交付方案的设计和实现。

自己动手写Docker

陈显鹭　王炳燊　秦妤嘉◎著



電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内容简介

本书在详细分析 Docker 所依赖的技术栈的基础上，一步一步地通过代码实例，让读者可以自己循序渐进地用 Go 语言构建出一个容器的引擎。不同于其他 Docker 原理介绍或代码剖析的书籍，本书旨在提供给读者一条动手路线，一步一步地实现 Docker 的隔离性，构建 Docker 的镜像、容器的生命周期及 Docker 的网络等。本书涉及的代码都托管在 GitHub 上，读者可以对照书中的步骤从代码层面学习构建流程，从而精通整个容器技术栈。本书也对目前业界容器技术的方向和实现做了简单介绍，以加深读者对容器生态的认识和理解。

本书适合对容器技术已经使用过或有一些了解，希望更深层次掌握容器技术原理和最佳实践的读者。

图书在版编目（CIP）数据

自己动手写Docker / 陈显鹭，王炳燊，秦妤嘉著. —北京：电子工业出版社，2017.7

ISBN 978-7-121-31786-6

Ⅰ. ①自… Ⅱ. ①陈… ②王… ③秦… Ⅲ. ①Linux操作系统－程序设计 Ⅳ. ①TP316.85

中国版本图书馆CIP数据核字（2017）第124163号

策划编辑：张春雨
责任编辑：徐津平
印　　刷：三河市鑫金马印装有限公司
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　　邮编：100036
开　　本：787×980　1/16　印张：13.25　字数：269千字
版　　次：2017年7月第1版
印　　次：2017年7月第1次印刷
定　　价：65.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

序

我是阿里云容器服务团队的架构师易立，很荣幸为这本书作序。

当显鹭等几位同学跟我谈起他们想写一本介绍如何从头打造一个 Docker 引擎的书时，我有些担心这样的内容是不是太小众，毕竟绝大多数读者都是 Docker 的使用者而非开发者。然而读完样章，看到这三位同学笔下翔实的内容，文中透出的热情和自信打消了我的顾虑。

Docker 是技术圈中的当红小鲜肉。自从 2013 年横空出世以来，迅速在开发者社区流行开来。在 2016 年 9 月，Docker 镜像在 Docker Hub 的总下载量就已经超过了 60 亿次，并且以每 6 周 10 亿次的速度迅速增长。

大家都知道 Docker 技术脱胎于 Linux Container（LXC）技术，在 LXC 的发展过程中，IBM、Google、Red Hat、Canonical 等技术巨擘做出了众多的贡献。然而，Docker 到底有什么魔力，能够在这么短的时间之内就风靡了整个技术圈呢？

Docker 公司的创始人兼 CTO——Solomon Hykes，有机地把一系列技术 Cgroups、Namespace 和 UnionFS 整合起来，极大地降低了容器技术的复杂度，提升了开发者的用户体验。他敏锐地预测到，一旦标准化容器技术最终出现，整个技术行业将会受到深远的影响。Docker 公司开源了 Docker Engine，定义了一个以容器镜像为标准的应用打包格式，并且建立 Docker Hub 服务进行镜像分发和协作。这些举措迅速创建了一个良好的社区和合作伙伴生态圈，包含 AWS、Google、Microsoft、IBM 和国内的众多公司。在短短几年的时间内，Docker 几乎成为了容器技术的代名词。

“得标准者得天下”，容器底层标准化之争风云再起。2014 年年底，CoreOS 推出 rkt 容器引擎，试图挑战 Docker 另立标准。Docker 在 2015 年 6 月宣布成立 OCI（Open Container Initiative）组织作为 Linux 基金会的协作项目，并将其容器标准和 runtime 参考实现（runC）贡献出来，旨在围绕容器格式和运行时制定一个开放的工业化标准。这一举措化解了社区在容器标准上的第一次分歧。

随着容器技术的快速发展，技术生态逐渐从围绕单机环境构建和运行容器化应用，发展为支持大规模容器编排技术。云平台成为了分布式网络操作系统，而容器成为了“进程”执行单元，可以动态地运行在不同宿主机环境中。其中，Kubernetes、Mesos、Docker 诸强争霸，各有所长。2016 年 6 月，Docker 宣布开始在 Docker Engine 中内置 Swarm mode，这极大地简化了容器编排的复杂性，但也遭到了社区的强烈反对。Google 发起 CRI（Container Runtime Interface，容器运行时接口）项目，通过 shim 的抽象层使得调度框架支持不同的容器引擎实现。Mesos 推出了 Unified Containerizer，以支持 Docker、Appc、OCI 等不同的镜像格式，而无须再依赖 Docker Engine。

面对这些挑战，2016 年 12 月 14 日，Docker 公司宣布将 Docker Engine 的核心组件 Containerd 捐赠到一个新的开源社区，任其独立发展和运营，目标是提供一个标准化的容器 runtime，其注重简单、健壮性和可移植性。由于 Containerd 只包含最基本的容器管理能力，因此上层框架可以有更大的灵活性来提供容器的调度和编排能力。阿里云、AWS、Google、IBM 和 Microsoft 作为 Containerd 的初始成员，为项目贡献力量。

在技术爆发的年代，新技术层出不穷，而快餐式的阅读和了解无法帮助我们梳理和把握发展的脉络。对一些核心技术既要知其然也要知其所以然，这样才能举一反三，对技术趋势建立起自己的理解和判断。了解容器基础知识，可以深入理解容器在进程管理、资源管理、安全隔离等方面与传统方式的不同，也有助于了解容器在网络、存储、安全等方面的特殊性。

最好的学习方式莫过于自己亲手实践。计算机界的泰斗 Andrew Tanenbaum 教授为教学而构建了 Minix，而这也启发了 Linus Torvalds 大神创造了 Linux。我们期待同学们能够从本书循序渐进的讲解中学习容器相关的技术细节，深入理解 Docker 的底层技术实现，围绕容器技术实现创造性的扩展和应用。

易立

2017 年 1 月

前言

为什么要写这本书

Docker 技术可谓是近年最火热的技术之一，铺天盖地的技术论坛和各种讲座，大家都在分享关于如何容器化及如何使用 Docker 优化自己运维和开发流程的经验。随着 Docker 技术的逐渐普及，使用 Docker 已经不再是一个难题。现在更加重要的是生产环境容器化的最佳实践，另外就是容器的编排框架之争。但是，对于技术人员来说，除去 Docker 外表的繁华，什么是容器，容器到底是怎么创建的，容器底层的技术探秘也是非常重要的。

我在 2014 年开始接触 Docker，经历了从最初的新奇——感叹竟然还有 Docker 这样的好工具，到逐渐熟悉 Docker 的各种功能，尝试在生产环境中使用 Docker 技术的过程。但是，每每被人问到："Docker 技术到底是怎么实现的呢？"我只能粗粗浅浅地说："Docker 是使用 Linux Kernel 的 Namespace 和 Cgroups 实现的一种容器技术。"那么，什么是 Namespace，什么是 Cgroups，Docker 是怎么使用它们的，容器到底是怎么一步步被创建出来的？问到这些，我就会支支吾吾地不知所以。由此可见，了解容器技术的底层技术，然后明白它们是如何工作的，尤为重要，这些才是整个容器技术的基石，掌握了这些基石才能更加容易地向上攀登。

单单讲解底层的技术实现细节和源码解读是很枯燥的一件事，一般来说很难有耐心去一点点细读然后揣摩其中的奥妙，这样囫囵吞枣地过一遍技术细节，作用不大。因此，我便萌生了写一本《自己动手写 Docker》这样的书的想法。本书不去刻意讲解容器技术的细节，用到什么讲解什么，点到为止，更加细节的内容留给读者自己探索。通过阅读本书，可以一步步地了解容器技术的实现细节，更可以一步步地用自己的代码去实现它。本书最大的乐趣莫过于用自己最新了解到的知识去动手实现自己的容器。由此可以进一步打开你进入容器技术社区的大门。

本书的内容

本书的目的是去引导读者通过学习容器技术的实现细节，一步步去构建一个简单的容器。从这个过程中，了解整个容器技术领域和实现细节。本书注重原理的讲解与实践，每一部分都会有详细的代码解析，力争用最少、最精简的代码，帮助读者构建自己的容器。

本书的内容主要分为“容器与开发语言”“基础技术”“构造容器”“构造镜像”“构造容器进阶”“容器网络”“高级实践”这 7 章。

- 容器与开发语言：主要介绍 Docker 的基本功能和特点，并且对后面即将使用的 Go 语言做一个简单的介绍。
- 基础技术：主要介绍实现容器的底层技术，如 Namespace、Cgroups、Union File System。每一节都会有文字性介绍，并且附有一个简短的小例子程序，介绍在容器上是如何使用这项技术的，方便读者清晰地理解各个技术点在容器上的作用。
- 构造容器：使用前面两章介绍的基础技术，构造一个最简单的容器环境，会将整体实现细节及代码解析一点点展现，直接使用前面介绍的基础技术，从而更加有实战感。
- 构造镜像：使用 2.3 节介绍的分层文件系统技术，构建一个简单的容器镜像，体现容器镜像的分层思想。
- 构造容器进阶：更加贴近真实的容器实现，在原来的基础上，增加更丰富的功能。通过这一章的学习，读者可以更好地了解各种技术是如何整合在一起去实现容器整体功能的。
- 容器网络：除实现一个容器环境之外，这一章还会讲解如何使自己的容器和宿主机通信，以及如何让不同的容器之间进行通信，更加贴近真实环境。
- 高级实践：使用自己编写的容器，运行一些通用程序，验证容器的可用性。此外，本章还介绍了目前 Docker 使用的容器运行引擎，以及目前容器运行态引擎的概况。

适用读者

- 希望更加深入地了解容器技术的读者。
- 已经使用过 Docker，希望探查细节的读者。
- Go 语言程序员，了解如何使用 Go 语言来编写自己的容器。
- 容器技术爱好者。

如何阅读

由于本书的定位是自己动手写 Docker，侧重于实战，因此仅仅纸面阅读是无法体会所有要点的。所有的源码均托管在 http://github.com/xianlubird/mydocker，您可以在这里下载到本书的所有源码。每一章节都会有一个对应的 tag，建议您在阅读书中讲解内容的同时，也将代码下载下来，在本机尝试运行，了解整个代码的运行流程。我们非常欢迎向本项目提交 Pull Request，在阅读思考中不断交流学习。

关于勘误

由于时间和水平都比较有限，因此本书难免会存在一些纰漏和错误。如果读者发现了问题，请及时与我们联系，我们也会在后面的版本中加以改正，邮箱是 xianlubird@gmail.com。非常希望与大家共同学习容器技术。

致谢

最后，向本书编写过程中给予我们巨大帮助的人们表示诚挚的感谢。感谢女友的支持，没有她包揽所有家务，我不会有大量空余时间编写此书。感谢同事姜继忠和谢瑶瑶对于本书 containerd 和 Kubernetes 相关章节内容的支持。感谢阿里云容器服务团队在本书编写期间给予的理解与包容。最后，感谢张春雨编辑，是他的帮助与支持才使得本书由一个想法变成了实体，展现给各位读者。

陈显鹭

读者服务

轻松注册成为博文视点社区用户（www.broadview.com.cn），扫码直达本书页面。

- **下载资源**：本书如提供示例代码及资源文件，均可在 下载资源 处下载。
- **提交勘误**：您对书中内容的修改意见可在 提交勘误 处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **交流互动**：在页面下方 读者评论 处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：*http://www.broadview.com.cn/31786*

目录

第1章 容器与开发语言

1.1 Docker

最近一段时间，云计算领域最火的莫过于“容器”一词。提到容器，就不得不提 Docker，可以说 Docker 已经成为了容器的代名词。那么，什么是 Docker？Docker 又能做什么呢？本章我们就来简单介绍一下 Docker。

1.1.1 简介

Docker 是一个开源工具，它可以将你的应用打包成一个标准格式的镜像，并且以容器的方式运行。Docker 容器将一系列软件包装在一个完整的文件系统中，这个文件系统包含应用程序运行所需要的一切：代码、运行时工具、系统工具、系统依赖，几乎有任何可以安装在服务器上的东西。这些策略保证了容器内应用程序运行环境的稳定性，不会被容器外的系统环境所影响。

图 1.1 是一个容器镜像的结构图。从中可以看到，镜像可以把系统级依赖都打包成一个文件，所有的容器会共享一个 Kernel，因此在同一个 Kernel 下可以运行各种 Linux 发行版的容器。

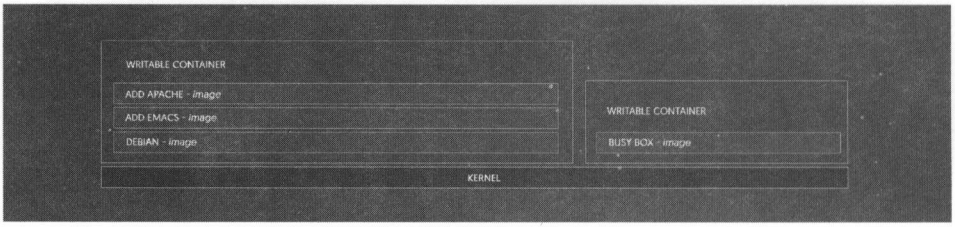


图 1.1

Docker 容器具有以下 3 个特点。

- 轻量级：在同一台宿主机上的容器共享系统 Kernel，这使得它们可以迅速启动而且占用内存极少。镜像是以分层文件系统构造的，这可以让它们共享相同的文件，使得磁盘使用率和镜像下载速度得到提高。
- 开放：Docker 容器基于开放标准，这使得 Docker 容器可以运行在主流 Linux 发行版和 Windows 操作系统上。
- 安全：容器将各个应用程序隔离开来，这给所有的应用程序提供了一层额外的安全防护。

1.1.2 容器和虚拟机比较

容器和虚拟机同样有着资源隔离和分配的优点，但是由于其架构的不同，容器比虚拟机更加便携和高效。

虚拟机包含用户的程序，必要的函数库和整个客户操作系统，所有的这些差不多需要占用好几个 GB 的空间。图 1.2 为虚拟机的架构图。

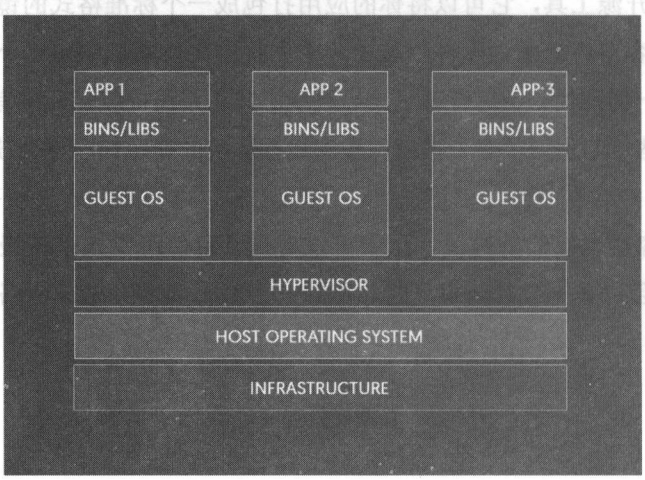


图 1.2